Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2
Технології розробки програмного забезпечення
«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Виконала:
студент групи ІА-24
Красношапка Р. О.
Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Тема: ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ

Мета: Розробити діаграми варіантів використання та класів.

Короткі теоретичні відомості

Діаграма варіантів використання (Use Case Diagram)

Діаграма варіантів використання описує функціональні можливості системи з точки зору її користувачів (акторів).

- **Елементи**:
  - **Актори**: користувачі системи (люди, пристрої чи інші системи).
  - **Варіанти використання (Use Cases)**: функції чи послуги, які система надає акторам.
  - **Зв'язки**: відображають взаємодію акторів із варіантами використання.
- **Призначення**: допомагає зрозуміти, що система повинна робити, і визначити її функціональність.

Сценарії варіантів використання (Use Case Scenarios)

Сценарій варіанту використання описує конкретну послідовність дій між актором і системою для досягнення певної цілі.

- **Основні компоненти**:
  - Назва сценарію.
  - Опис: короткий огляд того, що відбувається.
  - Основний потік: стандартна послідовність дій.
  - Альтернативні потоки: відхилення від основного сценарію.
  - Попередні умови: що має бути виконано перед початком сценарію.
  - Результат: стан системи після виконання сценарію.

Діаграми UML (Unified Modeling Language)

UML — це мова моделювання для створення візуальних діаграм, які описують різні аспекти системи.

- **Основні типи діаграм**:

1. **Структурні**: діаграми класів, компонентів, об'єктів, розгортання.
2. **Поведінкові**: діаграми варіантів використання, активності, послідовності, станів.
3. **Інтеракційні**: діаграми комунікації, часові діаграми.

- **Призначення**: полегшення аналізу, дизайну та документування системи.

Діаграма класів (Class Diagram)

Діаграма класів моделює структуру системи, відображаючи класи, їх атрибути, методи та взаємозв’язки.

- **Елементи**:
  - **Класи**: описують об'єкти (атрибути та методи).
  - **Зв’язки**:
    - Асоціація (association).
    - Агрегація (aggregation).
    - Композиція (composition).
    - Наслідування (inheritance).
  - **Мультиплікатори**: показують кількість об’єктів у зв’язку.
- **Призначення**: деталізує статичну структуру системи.

Концептуальна модель системи

Концептуальна модель системи — це абстрактне уявлення про основні об’єкти домену і зв’язки між ними.

- **Компоненти**:
  - Об'єкти/класи: реальні чи абстрактні сутності, що мають атрибути та операції.
  - Атрибути: властивості об'єктів.
  - Зв’язки: взаємодії між об'єктами.
- **Призначення**: допомагає зрозуміти бізнес-логіку та основні взаємозв'язки в системі.
- **Застосування**: використовується на початкових етапах проєктування для формування загального уявлення про систему.

Хід роботи

Project Management software (proxy, chain of responsibility, abstract factory, bridge, flyweight, client-server) Програмне забезпечення для управління проектами повинно мати наступні функції: супровід завдань/вимог/проектів, списків команд, поточних завдань, планування за методологіями agile/kanban/rup (включаючи дошку завдань, ітерації тощо), мати можливість прикріплювати вкладені файли до завдань та посилатися на конкретні версії програми, зберігати виконувані файли для кожної версії.

Діаграми варіантів використання (Use-case diagram)

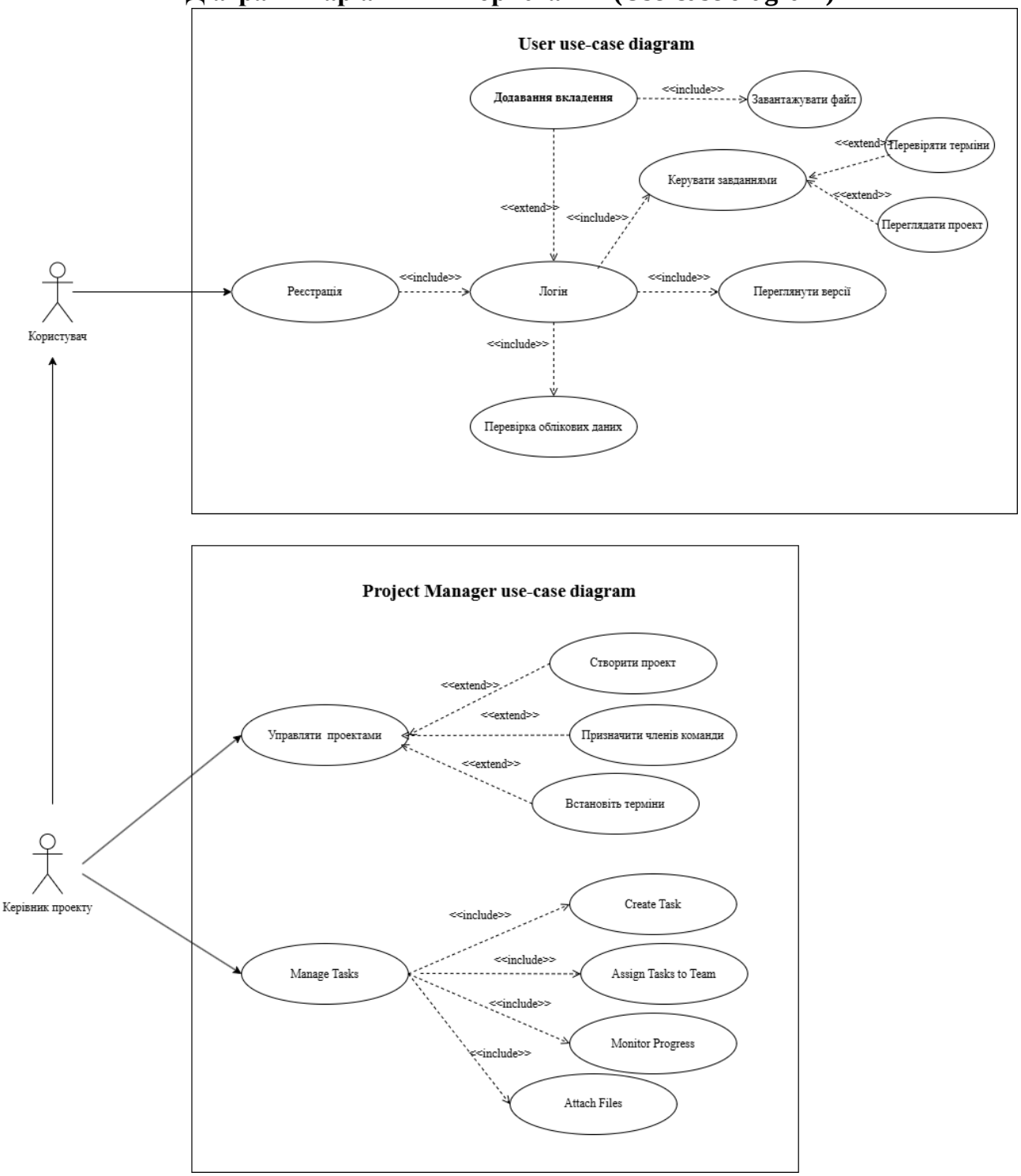

Рис. 1.1 - Use-case diagram

На діаграми випадків використання (Use Case Diagrams), які ілюструють функціональність програмного забезпечення для управління проєктами. Діаграми деталізують функції, доступні як для звичайних користувачів, так і для керівників проєктів, що дозволяє систематизувати процеси керування завданнями, проєктами та версіями.

Функціональність для звичайного користувача

На діаграмі, що описує роль звичайного користувача, основними випадками використання є:

- **Реєстрація**: передбачає створення нового облікового запису. Цей процес включає перевірку облікових даних для забезпечення безпеки та унікальності профілю.
- **Логін**: забезпечує авторизацію користувача через перевірку його облікових даних.
- **Додавання вкладень**: користувач може завантажувати файли до системи. Ця функція розширюється конкретним випадком "Завантажити файл".
- **Керування завданнями**: користувач отримує доступ до базових операцій із завданнями, зокрема перевірки термінів і перегляду проєктів.
- **Керування версіями**: дозволяє відслідковувати зміни та управляти різними версіями проєкту.

Функціональність для керівника проєкту

На діаграмі для керівника проєкту представлено більш розширений набір функцій:

- **Керування проєктами**:
    - Створення нового проєкту.
    - Призначення членів команди, що відповідає за розподіл завдань серед учасників.
    - Встановлення термінів виконання проєкту, що дозволяє організувати процес роботи.
- **Керування завданнями**:
    - Створення нових завдань.
    - Призначення завдань членам команди.
    - Моніторинг прогресу для контролю за виконанням поставлених задач.
    - Додавання файлів, необхідних для виконання роботи.

Типи зв’язків на діаграмах

1. **Include**: зв'язок, що вказує на обов’язкове включення одного випадку використання до іншого. Наприклад, процес авторизації (Логін) завжди включає перевірку облікових даних.

2. **Extend**: зв'язок, який визначає додаткову функціональність, що може бути виконана за необхідності. Наприклад, керування проєктами розширюється можливостями створення проєкту чи встановлення термінів.

Загальний висновок

Ці діаграми чітко демонструють розподіл функціональних обов’язків між кінцевими користувачами та керівниками проєктів. Для користувачів доступний базовий набір функцій, спрямованих на виконання окремих завдань, тоді як керівники проєктів мають розширені можливості для управління проєктами, завданнями та командою. Такий підхід забезпечує зручність, гнучкість і масштабованість системи, сприяючи ефективній організації робочого процесу.

Варіанти використання (прецеденти)

1) Додавання та редагування завдань у проекті

Передумови: Користувач авторизований у системі
Постумови: Завдання успішно додано або відредаговано у проекті
Взаємодіючі сторони: Користувач, система управління проектами
Короткий опис: Цей варіант описує процес додавання нового завдання або редагування існуючого завдання в проекті.
Основний потік подій:

1. Користувач відкриває проект у системі управління проектами.
2. Користувач натискає кнопку "Додати завдання" або обирає існуюче завдання для редагування.
3. Користувач заповнює необхідні поля, такі як опис, пріоритет, термін виконання та відповідальні особи.
4. Користувач натискає кнопку "Зберегти". Система зберігає завдання і підтверджує успішне додавання/редагування.
5. Користувач отримує сповіщення про успішне виконання дій.
   • Винятки:

Виняток №1: Пропущені обов'язкові поля. Якщо користувач не заповнив усі обов'язкові поля, система відображає відповідне повідомлення з деталями про помилку.

2) Прикріплення файлів до завдань

Передумови: Завдання вже існує у проекті
Постумови: Файл успішно прикріплений до завдання
Взаємодіючі сторони: Користувач, система управління проектами

Короткий опис: Даний варіант описує процес прикріплення файлів до завдання для зручності доступу та обміну інформацією.
Основний потік подій:

1. Користувач відкриває завдання у проекті.
2. Користувач натискає кнопку "Прикріпити файл".
3. Відкривається діалогове вікно для вибору файлу з комп'ютера.
4. Користувач вибирає файл і підтверджує вибір.
5. Система прикріплює файл до завдання та підтверджує успішне прикріплення.
   Винятки:
   • **Виняток №1:** Файл занадто великого розміру. Якщо розмір файлу перевищує максимальний допустимий, система виводить відповідне повідомлення про помилку.

3) Перегляд версій проекту

Передумови: Проект має кілька версій
Постумови: Користувач отримує інформацію про доступні версії проекту
Взаємодіючі сторони: Користувач, система управління проектами
Короткий опис: Цей варіант описує процес перегляду доступних версій проекту для можливості вибору та роботи з конкретною версією.
Основний потік подій:

1. Користувач відкриває проект у системі управління проектами.
2. Користувач обирає вкладку "Версії".
3. Система відображає список доступних версій проекту з інформацією про номер версії, дату випуску та примітки.
4. Користувач може вибрати конкретну версію для перегляду або завантаження.
   Винятки:
   • **Виняток №1:** Відсутні версії проекту. Якщо версії не знайдено, система відображає повідомлення про відсутність версій.

UML діаграма класів

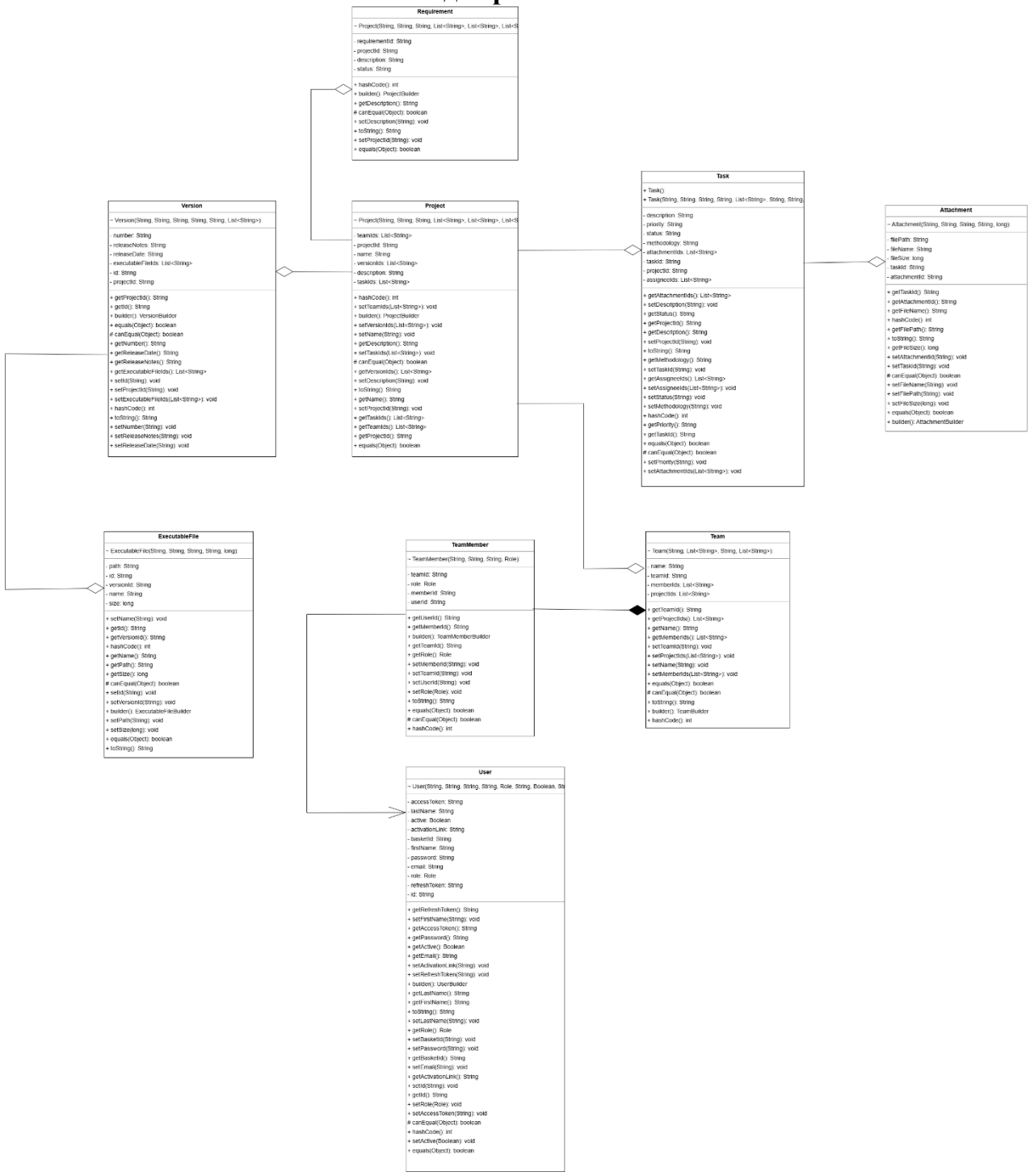

Рис. 2.1 – UML діаграма класів

На UML-діаграмі класів описано модель управління проєктами, завданнями та учасниками команди. Діаграма структурно представляє основні класи системи та їх взаємозв’язки, що дозволяє зрозуміти організацію даних і взаємодій у системі.

Основні класи системи

1. Project (Проєкт)

Цей клас є центральним у системі, оскільки описує проєкт, його властивості та пов’язані елементи.

- **Поля**:
    - id: унікальний ідентифікатор проєкту.
    - name: назва проєкту.
    - tasks: список завдань, пов’язаних із проєктом.
    - team: посилання на команду, яка працює над проєктом.
    - requirements: список вимог до проєкту.
- **Методи**: передбачають отримання, зміну та перевірку властивостей проєкту.
- **Зв’язки**:
    - Має зв’язок "один до багатьох" із класами Task (Завдання), Team (Команда) та Requirement (Вимога).

2. Task (Завдання)

Цей клас відповідає за опис завдань, що входять до складу проєкту.

- **Поля**:
    - id: унікальний ідентифікатор завдання.
    - name: назва завдання.
    - status: статус виконання завдання.
    - attachments: список вкладень, пов’язаних із завданням.
- **Методи**: дозволяють керувати статусом завдання, додавати вкладення та отримувати необхідну інформацію.
- **Зв’язки**:
    - Зв’язок із Attachment (один до багатьох) для додавання файлів.
    - Пов’язаний із Project і TeamMember.

3. Team (Команда)

Цей клас представляє команду, що працює над проєктом.

- **Поля**:
    - id: унікальний ідентифікатор команди.
    - name: назва команди.

    - members: список учасників команди.
- **Методи**: відповідають за додавання, видалення та управління учасниками.
- **Зв’язки**:
    - Має зв’язок "один до багатьох" із TeamMember.
    - Пов’язаний із Project.

4. TeamMember (Учасник команди)

Цей клас описує членів команди.

- **Поля**:
    - id: унікальний ідентифікатор учасника.
    - name: ім’я учасника.
    - role: роль у команді.
- **Методи**: включають управління даними про учасника.
- **Зв’язки**:
    - Пов’язаний із Team та Task.

5. Requirement (Вимога)

Цей клас використовується для опису вимог до проєкту.

- **Поля**:
    - id: унікальний ідентифікатор вимоги.
    - name: назва вимоги.
    - description: детальний опис.
- **Зв’язки**:
    - Має зв’язок із Project.

6. Attachment (Вкладення)

Цей клас представляє файли, пов’язані із завданнями.

- **Поля**:
    - id: унікальний ідентифікатор файлу.
    - name: назва файлу.
    - type: тип файлу (наприклад, зображення, документ).

    - path: шлях до файлу.
- **Зв’язки**:
    - Пов’язаний із Task.

7. ExecutorDetails (Деталі виконавця)

Цей клас використовується для зберігання інформації про виконавця.

- **Поля**:
    - name: ім’я виконавця.
    - position: посада виконавця.

Зв’язки між класами

- Project має зв’язки "один до багатьох" із класами Task, Team і Requirement.
- Task має зв’язки із Attachment (один до багатьох) і TeamMember.
- Team пов’язаний із TeamMember через зв’язок "один до багатьох".
- Requirement пов’язаний із Project, забезпечуючи опис специфікацій.

Зображення структури бази даних

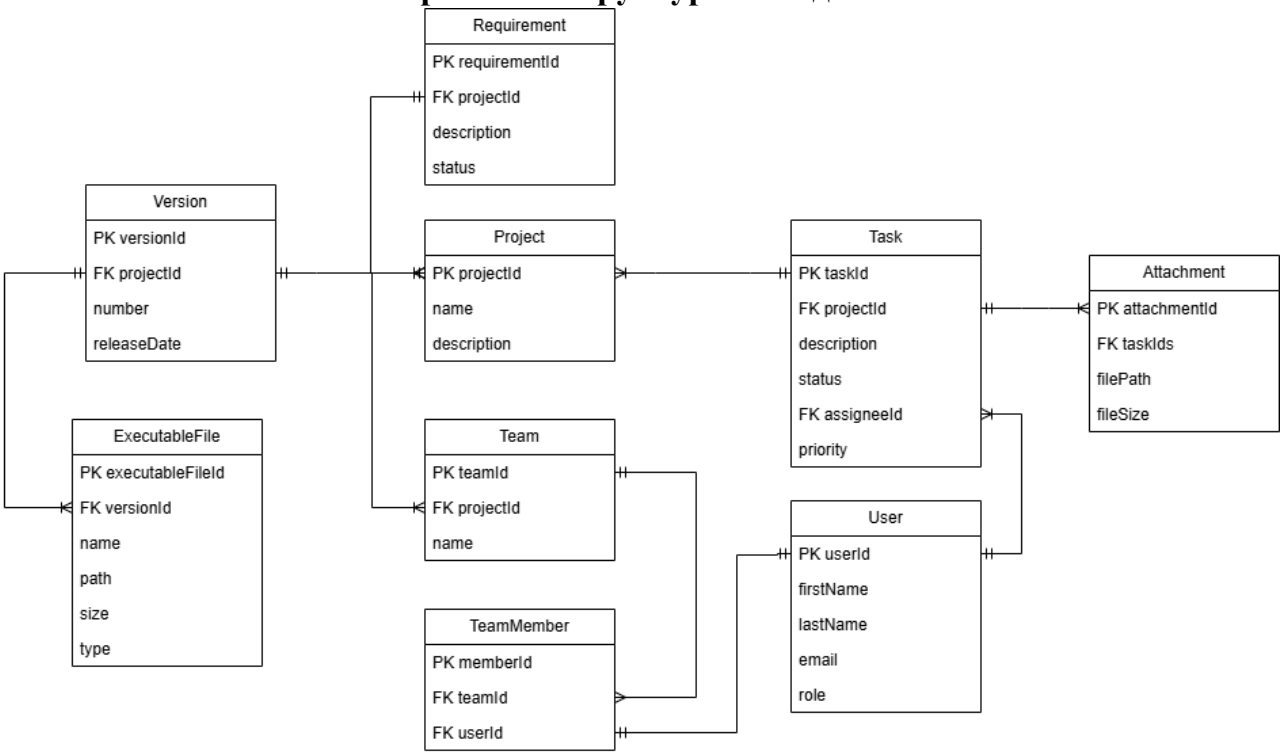

Рис. 3.1 – Структура бази даних

Ця діаграма бази даних описує структуру даних для програмного забезпечення управління проєктами, охоплюючи ключові аспекти: проєкти, задачі, команди, учасників, вимоги, версії, файли та вкладення. Нижче наведено опис основних таблиць і їх взаємозв’язків.

Основні таблиці:

1. Project (Проєкт)

Ця таблиця зберігає основну інформацію про проєкти.

- **Поля:**
  - projectId (PK): унікальний ідентифікатор проєкту.
  - name: назва проєкту.
  - description: опис проєкту.

2. Requirement (Вимоги)

Описує вимоги, пов’язані з конкретними проєктами.

- **Поля:**
  - requirementId (PK): унікальний ідентифікатор вимоги.
  - projectId (FK): зв’язок із таблицею **Project**.

- description: опис вимоги.
- status: статус вимоги (наприклад, "в процесі", "завершено").

3. Version (Версії)

Зберігає інформацію про релізи та версії проєктів.

- **Поля:**
  - versionId (PK): унікальний ідентифікатор версії.
  - projectId (FK): зв’язок із таблицею **Project**.
  - number: номер версії.
  - releaseDate: дата релізу.

4. ExecutableFile (Виконувані файли)

Таблиця для зберігання інформації про файли, пов’язані з конкретними версіями проєктів.

- **Поля:**
  - executableFileId (PK): унікальний ідентифікатор файлу.
  - versionId (FK): зв’язок із таблицею **Version**.
  - name: назва файлу.
  - path: шлях до файлу.
  - size: розмір файлу.
  - type: тип файлу.

5. Task (Задачі)

Ця таблиця зберігає дані про задачі в межах проєктів.

- **Поля:**
  - taskId (PK): унікальний ідентифікатор задачі.
  - projectId (FK): зв’язок із таблицею **Project**.
  - description: опис задачі.
  - status: статус задачі.
  - assigneeId (FK): посилання на виконавця задачі в таблиці **User**.
  - priority: пріоритет задачі.

6. Attachment (Додатки)

Використовується для зберігання додаткових файлів, прикріплених до задач.

- **Поля:**
    - attachmentId (PK): унікальний ідентифікатор додатка.
    - taskId (FK): зв’язок із таблицею **Task**.
    - filePath: шлях до файлу.
    - fileSize: розмір файлу.

7. Team (Команда)

Містить дані про команди, які працюють над проєктами.

- **Поля:**
    - teamId (PK): унікальний ідентифікатор команди.
    - projectId (FK): зв’язок із таблицею **Project**.
    - name: назва команди.

8. TeamMember (Учасники команди)

Зберігає інформацію про членів команд.

- **Поля:**
    - memberId (PK): унікальний ідентифікатор учасника.
    - teamId (FK): зв’язок із таблицею **Team**.
    - userId (FK): зв’язок із таблицею **User**.

9. User (Користувачі)

Зберігає дані про користувачів, які беруть участь у проєктах.

- **Поля:**
    - userId (PK): унікальний ідентифікатор користувача.
    - firstName: ім’я користувача.
    - lastName: прізвище.
    - email: електронна адреса.
    - role: роль користувача (наприклад, менеджер, розробник).

Зв’язки між таблицями:

- Таблиця **Project** пов’язана з **Requirement**, **Version**, **Team** і **Task** через зовнішні ключі.

- **Task** має зв’язок із **Attachment** і **User** (виконавець).
- **Version** зв’язана з **ExecutableFile**, що зберігає файли релізів.
- **Team** пов’язана із **TeamMember**, які, у свою чергу, посилаються на **User**.

Ця структура забезпечує комплексне управління проєктами, від планування задач і відстеження вимог до роботи з командами, версіями програмного забезпечення і файлами. Вона добре підходить для систем, які вимагають гнучкості, масштабованості та інтеграції між компонентами.

Посилання на репозиторій:

https://github.com/QUIRINO228/projectManagmentSoftware

Висновок: У процесі розробки системи управління проєктами використано різноманітні засоби моделювання для забезпечення чіткого розуміння структури та функціональності системи:

Діаграми варіантів використання допомагають ідентифікувати ключові ролі користувачів (наприклад, менеджери, розробники) і їх взаємодію із системою. Вони визначають основні функції, такі як створення проєктів, планування задач, управління командами та робота з файлами.

Сценарії варіантів використання деталізують поведінку системи в рамках кожного випадку використання. Це дозволяє краще зрозуміти послідовність дій користувача та відповідь системи, що сприяє виявленню можливих проблем на етапі проектування.

Діаграми UML надають графічне представлення структури та динаміки системи, зокрема її класів, об’єктів та їх взаємодії. Вони є важливими інструментами для комунікації між членами команди.

Діаграми класів формують основу концептуальної моделі системи, описуючи її основні елементи (класи, атрибути, методи) та зв’язки між ними. Вони забезпечують цілісне уявлення про структуру даних і функціональні можливості системи.

Концептуальна модель системи узагальнює всі ключові аспекти функціональності, забезпечуючи основу для подальшого детального проектування і розробки. Вона дозволяє зберегти баланс між простотою та функціональністю, враховуючи потреби кінцевих користувачів.

Ці засоби моделювання в комплексі забезпечують послідовне, структуроване та прозоре проектування системи, мінімізують ризики помилок та сприяють її успішній реалізації.